SYSGREEN

```
poe_main_call_par_fu
(argc,argv) {
... res=par_fu(...);
poe_return_result(res); }
```

24
14
28

SYSORANGE

```
poe_main_call_par_fu
(argc,argv) {
... res=par_fu(...);
poe_return_result(res); }
```

24
16
28

SYSRED

```
original_program()
...
poe_call_func("par_fu", args, res, par_opts);
...
poe_call_func(...)
{...spawn("poe poe_call_main_par_fu",
args,par_opts)...}
```

IPC

poe poe_main_call_par_fu",args,par_opts

18
20
26
22
10
12

SYSYELLOW

```
poe_main_call_par_fu(argc,argv) {
... res=par_fu(...); poe_return_result(res); }
```

24
28

FIG. 1

210

SERIALIZE ARGUMENTS FOR PARALLEL METHOD

220

CALL POE THROUGH A STANDARD FUNKTION
ENCAPSULATING ALL PARALLELIZATION CONTROL

230

GENERATE A NEW POE PROCESS

240

INVOKE PARALLEL METHOD MULTIPLY

250

RETURN SERIALIZED RESULT TO POE PROCESS
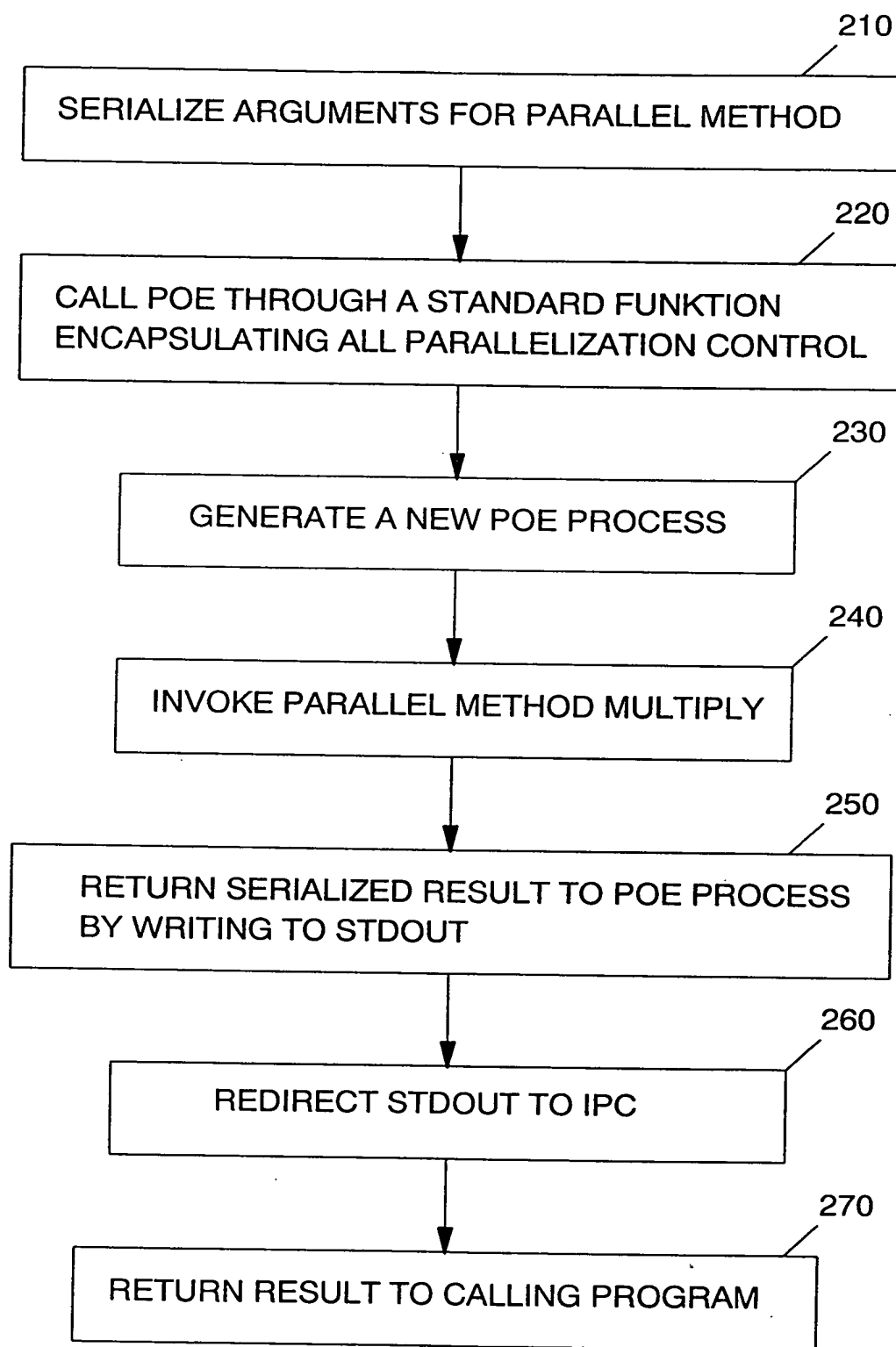BY WRITING TO STDOUT

260

REDIRECT STDOUT TO IPC

270

RETURN RESULT TO CALLING PROGRAM

FIG.2

	calling the parallel method (PM) (read column top down)	returning result from parallel method (PM) (read column bottom up)
Original program (OP) - user code	210: serialize arguments for PM 220: procedural poe call of PM with serialized arguments	280: deserialize result in result variable
Original Program (OP) - procedural POE call	221: derive name of parallel subprogram (PSP) from name of PM 222: set up IPC connection to poe process to be created 230: spawn POE process with name of PSP and serialized arguments	270 : move result from IPC connection to result variable
poe process	235: invoke PSB at least once	260: redirect standard output from PSP to IPC connection to OP
parallel subprogram	240: call PM with command line arguments as method arguments	250: return serialized result of PM to POE by writing it to standard output

Fig. 3